Um serviço de núcleo para informações sobre usuários

Diogo R. Olsen[*], Carlos A. Maziero

[1]Programa de Pós-Graduação em Informática
Pontifícia Universidade Católica do Paraná
Curitiba, PR – Brasil

diogoolsen@gmail.com.br, maziero@ppgia.pucpr.br

Resumo. *Em sistemas multi-usuários, o equilíbrio no uso dos recursos entre usuários é importante para evitar injustiças. Todavia, sistemas operacionais como o Linux não provêem uma forma eficiente de obter informações sobre o uso de recursos pelos usuários. Este artigo apresenta um serviço de núcleo e sua interface para prover informações sobre o uso de recursos por usuários e grupos. Os dados coletados podem ser consultados por outros sub-sistemas do núcleo e também por aplicativos de gerência fora dele, através da interface proposta.*

Abstract. *In multi-user systems, the good balance in resource usage among users is important to avoid unfairness situations. However, operating systems like Linux to not provide an efficient way to obtain information about resource usage by users. This paper presents a kernel service and its interface to provide information on how resources are allocated to users and user groups. Collected data can be used by other kernel sub-systems and also by management application in user level, through the proposed interface.*

1. Introdução

Em um sistema operacional multi-usuários, o equilíbrio no uso dos recursos entre os usuários é importante para evitar injustiças. Para gerenciar os recursos de um sistema, é necessária uma visão detalhada do uso dos mesmos, seja por processos ou por usuários. Sistemas como o Linux não provêem uma forma direta e eficiente de obter informações sobre o uso de recursos por seus usuários. Por exemplo, o pseudo-sistema de arquivos `/proc` [Faulkner and Gomes 1991], onde ferramentas como `ps` e `top` buscam seus dados, oferece apenas uma visão centrada em processos.

Este artigo apresenta um serviço de núcleo e sua respectiva interface para coletar e prover informações sobre o uso de recursos por usuários e grupos. São analisadas duas abordagens para a implementação do serviço de coleta de dados dentro do núcleo. Os dados coletados são consolidados e mantidos em estruturas que podem ser consultadas por sub-sistemas de gerência intra-núcleo. Além disso, esses dados são oferecidos ao espaço de processos (*user level*) através de sub-diretórios no sistema de arquivos `/proc`, onde podem ser utilizados por aplicativos de gerência.

O texto está estruturado da seguinte forma: a seção 2 apresenta conceitos relacionados ao uso de recursos em um sistema operacional, apresenta os mecanismos disponíveis no Linux e discute a contabilização do uso de recursos por usuários e grupos. A

[*]Bolsista de iniciação científica no programa PIBIC CNPq/PUCPR.

seção 3 apresenta a proposta deste trabalho, discute as abordagens de implementação e os resultados esperados. A seção 4 expõe a situação atual do trabalho e delineia os próximos passos. A seção 5 discute alguns trabalhos correlatos.

2. Uso de recursos do sistema

Em sistemas multi-usuários, é importante definir e aplicar políticas de gestão de recursos que garantam a justiça na distribuição dos mesmos entre os usuários do sistema, independente da quantidade de processos que cada um possui. Para isso, é necessário contabilizar o uso de recursos pelos usuários. Todavia, a maioria dos sistemas operacionais não provê uma forma direta e eficiente de obter informações sobre o uso de recursos pelos usuários, obrigando a coleta dessa informação por meios indiretos.

O procedimento de contabilização é relativamente simples para certos recursos, como o número de *threads* ou de arquivos abertos. Entretanto, algumas situações representam desafios consideráveis, como a contabilização de áreas de memória compartilhada e de bibliotecas dinâmicas [Emelianov et al. 2007].

Outro ponto que exige atenção são as várias identidades associadas aos processos no sistema. Por exemplo, o núcleo Linux suporta quatro identificadores de usuário e grupo para cada processo (definidos no arquivo `include/linux/sched.h`), que podem ter valores distintos em algumas situações. Esses identificadores são apresentados na tabela 1. Situação similar ocorre quando um usuário pertence a vários grupos: como contabilizar os recursos usados por aquele usuário entre os grupos a que pertence?

Tipo	Usuário	Grupo
Usuário/grupo real do processo	`uid`	`gid`
Usuário/grupo efetivo do processo	`euid`	`egid`
Usuário/grupo que iniciou o processo	`suid`	`sgid`
Usuário/grupo no sistema de arquivos	`fsuid`	`fsgid`

Tabela 1. Identificações de usuários e grupos no Linux.

2.1. Informações sobre uso de recursos no Linux

No linux existem várias maneiras de extrair informações sobre os recursos computacionais em uso; alguns exemplos são os programas `top`, `ps`, `vmstat` e `uptime`. A principal fonte das informações apresentadas por esses comandos é o pseudo sistema de arquivos `/proc` [Killian 1984, Faulkner and Gomes 1991], que oferece um grande número de informações sobre o sistema e os processos existentes, estruturadas na forma de arquivos e diretórios e acessíveis aos processos fora do núcleo. Além disso, o núcleo Linux estende esse pseudo sistema de arquivos para suportar operações de configuração do próprio núcleo. Algumas entradas típicas desse sistema de arquivos são:

- `/proc/cpuinfo`: contém informações sobre o número de processadores da máquina e as características técnicas de cada um;
- `/proc/modules`: indica os módulos atualmente carregados na memória do núcleo;
- `/proc/`*pid*`/cmdline`: indica a linha de comando usada para lançar o processo cujo identificador numérico é *pid*.

As informações contidas no sistema `/proc` são produzidas sob demanda, no momento em que cada arquivo é acessado. Esta característica do sistema `/proc` torna-o eficiente, pois a informação consolidada só é gerada se e quando necessário. Parte das informações oferecidas pelo sistema `/proc` é oriunda das estruturas de dados `task_struct` e `user_struct`, definidas no arquivo `include/linux/sched.h`:

- `task_struct`: estrutura do núcleo que define uma tarefa (processo ou *thread*). Cada processo do sistema possui uma instância desta estrutura, que armazena informações como: dono do processo, número de *threads*, status, prioridade, arquivos abertos, identificador (*pid*), processo pai, processos filhos, uso do processador, da memória e do sistemas de arquivos pelo processo, sinais e último contexto salvo.
- `user_struct`: define um usuário ativo no sistema, ou seja, que tenha ao menos um processo em seu nome. Cada processo está associado a um ou mais usuários ativos. Seus principais campos são: número de processos, número de arquivos abertos, número de sinais pendentes, identificador numérico de usuário e de grupo.

Ao analisar essas estruturas, percebe-se que há pouca informação consolidada por usuário ou por grupo. Sequer existe uma estrutura `group_struct`, para armazenar informações sobre grupos de usuários. Para obter informações consolidadas por usuário ou por grupo, torna-se necessário varrer continuamente a tabela de processos e computar os valores desejados. Outra possibilidade seria estender o código do núcleo para contabilizar o uso dos recursos pelos usuários e grupos, armazenando essas informações em novos campos na estrutura `user_struct` e em uma nova estrutura `group_struct`.

2.2. Informações mapeáveis por usuários e grupos

Nem toda informação sobre uso de recursos contida no núcleo pode ser diretamente contabilizada por usuários ou grupos. Algumas informações são facilmente quantificáveis, como o número de processos ou *threads* de cada usuário. Outras, como o consumo de memória, podem ser mais complexas. Grosso modo, as seguintes informações podem ser contabilizadas por usuários e/ou grupos de usuários:

- **Uso de memória**: volumes usados de memória virtual, real (RAM), compartilhada, tamanho total das áreas de código, dados, pilha, *heap* e *swap*, número de páginas limpas e sujas, número de faltas de página, número de alocações e liberações de memória dinâmica.
- **Uso de processador**: tempo total de processamento, número de trocas de contexto provocadas, fração do tempo em que processos do usuário ou grupo estiveram ativos.
- **Uso de disco**: espaço total usado pelo usuário, número de *i-nodes*, número de operações de leitura/escrita, número de blocos lidos/escritos.
- **Uso de rede**: número de pacotes e bytes recebidos ou enviados, número de *sockets* abertos, razão da banda de rede disponível utilizada, número de erros de envio de pacotes.
- **Uso de abstrações do sistema**: número de semáforos usados, número e tamanho total das áreas de memória compartilhada, número de pipes, número de filas de mensagens, número de descritores de arquivos abertos, números de processos e de *threads*, número de processos em cada estado (rodando, suspenso, etc), número de sinais recebidos ou enviados (por tipo de sinal).

Essa lista de parâmetros mensuráveis por usuário não pretende ser exaustiva.

3. Proposta

O objetivo principal deste trabalho é coletar e manter, dentro do núcleo do sistema Linux, informações sobre o uso corrente de recursos do sistema por usuários e grupos. Para tal é necessário definir as informações a coletar/manter, o mecanismo de coleta dessas informações, as estruturas de dados nas quais elas serão armazenadas e a interface de acesso às mesmas. Esta seção discute cada um destes aspectos.

3.1. Informações a coletar

Conforme discutido na seção 2.2, há várias informações sobre uso de recursos disponíveis no núcleo do sistema operacional que podem ser agregadas por usuários ou grupos. Para a construção do protótipo, serão inicialmente coletadas as informações descritas a seguir, para cada usuário e grupo ativos. Acredita-se que, uma vez construído o arcabouço para a coleta, armazenamento e oferta desses dados, novas informações poderão ser facilmente agregadas ao mesmo.

- **Processador**: número de processos e de *threads* em cada classe de escalonamento, percentual de uso do(s) processador(es), número de processos na fila de prontos em cada classe de escalonamento;
- **Memória**: tamanho total de memória virtual, real e *swap*, tamanho total das áreas compartilhadas, número de operações de *swap* solicitadas, número de páginas em bibliotecas, páginas sujas, páginas de código, páginas de dados e páginas de pilha;
- **Disco**: número de leituras/escritas de blocos, número de arquivos abertos;
- **Rede**: número de pacotes/bytes enviados e recebidos, número de sockets abertos.
- **Outros recursos**: número de filas de mensagens e semáforos em uso.

Os dados coletados serão armazenados dentro do núcleo, nas estruturas de dados `user_struct` (já existente, a ser estendida) e `group_struct` (a ser inteiramente definida). Esses dados estarão disponíveis dentro do núcleo, para os módulos ou *threads* de núcleo que tiverem interesse em usá-los. Além disso, serão exportados para o espaço de usuário (*userland*), conforme será descrito na seção 3.3. Cabe ressaltar que essas estruturas só estão definidas para usuários/grupos ativos no sistema, ou seja, usuários/grupos que tenham ao menos um processo em execução. Quando um usuário/grupo deixa de estar ativo, seus dados são eliminados.

3.2. Mecanismo de coleta

Existem duas possibilidades de implementação do mecanismo de coleta das informações sobre usuários e grupos: por *varredura periódica* das estruturas existentes ou por inserção de *pontos de coleta* no código do núcleo. Na primeira abordagem, as estruturas de dados existentes no núcleo são analisadas periodicamente, buscando informações que são consolidadas por usuários e por grupos e armazenadas nas estruturas descritas na seção anterior. Esta solução é atraente, pois não mexe no código do núcleo e pode ser implementada usando um módulo.

A varredura das estruturas de dados fica a cargo de uma *thread* de núcleo criada pelo módulo, que executa periodicamente de forma preemptiva, sem bloquear o restante

do núcleo. Todavia, esta solução não permite obter todas as informações desejadas, como é o caso do número de pacotes de rede enviados/recebidos por um usuário. Além disso, a definição do período de varredura é crítica: um período muito pequeno vai gerar uma carga de trabalho significativa para o núcleo, enquanto um período muito longo pode fazer com que recursos usados e liberados dentro de um mesmo período deixem de ser contabilizados.

A segunda abordagem consiste em inserir no núcleo do sistema, rotinas de coleta de informações. Por exemplo, inserir dentro da função `vmalloc` (em `linux/mm/vmalloc.c`) código para contabilizar o tamanho da região de memória alocada ao usuário/grupo que a solicitou. Para cada recurso a contabilizar, devem ser identificados os pontos de coleta das informações e inserir o código necessário em cada ponto. Em alguns casos podem ser usados mecanismos de interceptação disponíveis no núcleo do Linux, como os *Linux Security Modules* [Wright et al. 2002]. Apesar desta abordagem ser mais precisa, ela exige a intervenção em muitos pontos do núcleo, tornando sua implementação e depuração significativamente mais complexas, além de ser menos portável entre versões do núcleo.

3.3. Interface de acesso

Os dados coletados e mantidos nas estruturas `user_struct` e `group_struct` podem ser acessados dentro do núcleo diretamente, onde podem ser usados por mecanismos de controle de uso. Para que esses dados estejam disponíveis a aplicações de monitoração e gerência no espaço de usuário, eles devem ser exportados para fora do núcleo. Isso poderia ser feito através de chamadas de sistema definidas especificamente para esse fim, ou através de interfaces já presentes no núcleo, como o pseudo sistema de arquivos `/proc` [Killian 1984].

A criação de novas chamadas de sistema modifica a API do núcleo, o que não é desejável do ponto de vista das aplicações já existentes. Por isso, decidiu-se criar novas entradas no sistema `/proc` para oferecer os dados coletados aos processos. De forma similar ao mecanismo usado para fornecer informações sobre processos (disponível em `/proc/`*pid*`/`), dois novos sub-diretórios são definidos: `/proc/user/` e `/proc/group/`:

- `/proc/user/`*uid*`/`: contém entradas cujo conteúdo informa sobre o uso de recursos pelo usuário cujo identificador é *uid*. As entradas atuais desse diretório são:
  - `mem`: arquivo contendo informações sobre uso de memória;
  - `cpu`: arquivo contendo informações sobre uso de processador;
  - `procs`: estatísticas sobre uso de processos e *threads*;
  - `files`: arquivo contendo informações sobre uso de arquivos;
  - `fd/`: sub-diretório contendo os arquivos abertos pelo usuário;
- `/proc/group/`*gid*`/`: contém entradas cujo conteúdo informa sobre o uso atual de recursos pelo grupo de usuários cujo identificador é *gid*. O conteúdo desse diretório é similar ao dos diretórios de usuários.

Por exemplo, o conteúdo do arquivo `/proc/user/1005/mem` traz informações sobre o uso de memória pelos processos do usuário cujo *uid* é 1005:

```
VirtMemSize:    5848 kB
VirtMemRSS:     3268 kB
VirtMemData:    1960 kB
VirtMemStack:     84 kB
VirtMemText:     668 kB
VirtMemDLLs:    1784 kB
VirtMemShared:   354 kb
VirtMemDirty:     91 kb
VirtMemSwap:     132 kb
VirtMemFaults:    65
```

Da mesma forma, o arquivo `/proc/user/1005/proc` traz informações sobre processos e threads do usuário com *uid* 1005:

```
ProcNumProc:    21
ProcNumThread: 25
ProcStatus:     2 13 1 1 0 4  // R S D Z T W
```

4. Situação Atual

Até o momento foi construído um módulo de núcleo que implementa a coleta de informações por varredura das estruturas do núcleo (usando a abordagem de *varredura periódica* descrita na seção 3.2) e exporta os dados obtidos através de arquivo no diretório `/proc`. As informações atualmente oferecidas são um sub-conjunto daquelas definidas na seção 3.1.

Os próximos passos do projeto consistem em definir as formas de coleta das demais informações definidas na seção 3.1, definir a estrutura `group_struct` e implementar a coleta de dados através de *pontos de coleta* no núcleo. Para algumas informações, como o uso de memória, será necessário um estudo mais aprofundado sobre os mecanismos de gerência de memória virtual implementados no núcleo do Linux. Uma vez concluída a implementação do protótipo, serão definidos e realizados experimentos para medir a qualidade dos dados obtidos e o impacto dos mecanismos implementados no desempenho do núcleo.

5. Trabalhos correlatos

A pesquisa sobre instrumentação e coleta de informações sobre uso de recursos em sistemas operacionais é antiga [Saltzer and Gintell 1970]. Por exemplo, o artigo [Seltzer and Small 1997] propõe métodos para o auto-monitoramento do sistema operacional VINO, visando construir mecanismos de auto-adaptação do sistema operacional. Alguns sistemas operacionais, como o *Solaris*, oferecem algumas informações de monitoração consolidadas por usuários.

O trabalho [Starke et al. 2004] define um modelo de gerência de recursos no qual as políticas podem ser definidas por processos, grupos de processos, usuários e/ou grupos de usuários. O ambiente CKRM [Nagar et al. 2004] implementa uma gerência de recursos baseada em classes, onde usuários e grupos podem ser associados a classes. Mais recentemente, o escalonador de processos CFS (*Completely Fair Scheduler*) implementa essa mesma funcionalidade no núcleo Linux, mas somente para a alocação de processador.

6. Conclusão

Este artigo apresentou um trabalho em andamento que visa implementar uma infra-estrutura de coleta, armazenamento e oferta de informações sobre o uso de recursos do sistema operacional, consolidadas por usuários e por grupos de usuários. Essa infra-estrutura é composta por mecanismos de coleta de dados, estruturas de dados para seu armazenamento e uma interface para o acesso aos dados por processos no espaço de usuário. Esse conjunto de informações é importante para a construção de políticas de gestão de recursos mais elaboradas, que levem em conta a distribuição dos recursos entre os usuários.

No código fonte do núcleo Linux, dentro da definição da estrutura de dados `user_struc` (no arquivo `sched.h`), há um comentário sugestivo: *Some day this will be a full-fledged user tracking system...* Este comentário revela a preocupação dos desenvolvedores do núcleo Linux em oferecer um conjunto consistente de informações sobre usuários. Todavia, até este momento nenhum trabalho significativo foi realizado nesse sentido.

Este trabalho faz parte de um projeto mais amplo, envolvendo alunos de mestrado e doutorado, que visa a construção de mecanismos de gerência autonômica de recursos para sistemas operacionais multi-usuários.

Referências

Emelianov, P., Lunev, D., and Korotaev, K. (2007). Resource management: Beancounters. *Linux Symposium*, pages 285–297.

Faulkner, R. and Gomes, R. (1991). The process file system and process model in UNIX System V. In *USENIX Conference Proceedings*.

Killian, T. J. (1984). Processes as files. In *USENIX Software Tools Users Group Summer Conference*.

Nagar, S., van Riel, R., Franke, H., Seetharaman, C., Kashyap, V., and Zheng, H. (2004). Improving linux resource control using CKRM. In *Ottawa Linux Symposium*.

Saltzer, J. and Gintell, J. (1970). The instrumentation of Multics. *Communications of the ACM*, 13(8).

Seltzer, M. and Small, C. (1997). Self-monitoring and self-adapting operating systems. In *6th Workshop on Hot Topics in Operating Systems*.

Starke, M., Maziero, C., and Jamhour, E. (2004). Controle dinâmico de recursos em sistemas operacionais. In *Anais do I Workshop de Sistemas Operacionais da SBC*.

Wright, C., Cowan, C., Morris, J., Smalley, S., and Kroah-Hartman, G. (2002). Linux security modules: General security support for the Linux kernel. In *11th USENIX Security Symposium*.